AD-A218 890

DTIC FILE COPY

RADC-TR-89-292
Final Technical Report
December 1989

INTELLIGENT SIGNAL PROCESSING TECHNIQUES FOR MULTI-SENSOR SURVEILLANCE SYSTEMS

Syracuse University

Harvey Rhody, David Sher, James Modestino

DTIC
ELECTE
MAR 07 1990
S E D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

90 03 06 076

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-89-292 has been reviewed and is approved for publication.

APPROVED: Vincent C. Vannicola

VINCENT C. VANNICOLA
Project Engineer

APPROVED: Frank J. Rehm

FRANK J. REHM
Technical Director
Directorate of Surveillance

FOR THE COMMANDER: John A. Ritz

JOHN A. RTIZ
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (OCTS), Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-89-292
6a. NAME OF PERFORMING ORGANIZATION Syracuse University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (OCTS)
6c. ADDRESS (City, State, and ZIP Code) Division of Research Support and Administration Skytop Office Building Syracuse NY 13244-5300	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) OCTS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-88-D-0027
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62702F; PROJECT NO.: 4506; TASK NO.: 11; WORK UNIT ACCESSION NO.: PY

11. TITLE (Include Security Classification)
INTELLIGENT SIGNAL PROCESSING TECHNIQUES FOR MULTI-SENSOR SURVEILLANCE SYSTEMS

12. PERSONAL AUTHOR(S) Harvey Rhody, Rochester Institute of Technology, David Sher, State University of New York at Buffalo, James Modestino, Rensselaer Polytechnic Institute

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Apr 88 TO Apr 89	December 1989	64

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
09	04	13	Signal Processing; Surveillance Systems; Radar; Artificial Intelligence, (ATR)
09	04	14	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this project is to develop an analytical frame work to represent the modern multi-target, multi-sensor surveillance environment and to investigate the adaptation of intelligent signal processing algorithms to that application. The outcome is to be a road map for the development of the system elements and a plan for integrating them into a functional body.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Vincent Vannicola	22b. TELEPHONE (Include Area Code) (315) 330-4437 22c. OFFICE SYMBOL RADC (OCTS)

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Contents

Chapter 1

Introduction

This is the final report on a study of intelligent signal processing techniques for multi-sensor surveillance systems. The project was carried out under the RADC Expert Scientist and Engineering Program contract F30602-88-D-0027 Task Number A-8-1125.

1.1 Participants

The study was conducted by participants from the Rochester Institute of Technology, the State University of New York at Buffalo and Rensselaer Polytechnic Institute. The study was done over the period April 1988 to April 1989.

1.2 Background

Current surveillance systems must perform a variety of tasks within a complex real-time environment. Artificial intelligence (AI) techniques which have been developed for modern signal processing applications such as vision, image understanding and speech understanding combined with other AI techniques such as expert systems, knowledge representation, plan recognition, search and control offer ways to improve the performance of the next generation of surveillance systems.

Current surveillance systems make use of extensive signal processing for target detection, tracking and recognition. The signal processing has been

optimized for the given target/sensor combinations. It is likely that the next generation of systems will make use of these sensors and their processing algorithms but that the information will be used in a different way. The information from a number of sensors will be combined by a higher-level system to provide enhanced detection, tracking and recognition as well as the ability to recognize threats and characteristic or uncharacteristic behavior in complexes of targets.

The next generation of systems are likely to make use of a distributed set of sensors and processors. This approach offers the greatest modularity and flexibility in system development, deployment and maintenance. If effective custom systems can be constructed from a set of generic modules then there will be a very substantial savings over the cost of individual custom systems for a variety of applications.

Intelligent systems offer systematic processes to address problems such as sensor fusion, sensor coordination, threat assessment, decision analysis and resource allocation. All such tasks require that information be handled at a high level so that symbolic reasoning can be supported.

The behavior of distributed systems with significant numbers of interacting components is difficult to analyze and predict. Even if the individual system elements are well-understood, a system composed of many of them may exhibit new and unexpected modes. The inclusion of nonlinear processes, as decision processes must be, makes the theoretical analysis of such systems essentially impractical. Therefore the behavior and performance of distributed systems can be best evaluated by the use of prototypes and simulations.

The purpose of this project is to develop an analytical framework to represent the modern multi-target, multi-sensor surveillance environment and to investigate the adaptation of intelligent signal processing algorithms to that application. The outcome is to be a road map for the development of the system elements and a plan for integrating them into a functional body.

1.3 Symbolic *vs* Numerical Approaches

Modern surveillance systems of the radar/sonar variety are complex systems which historically have made use of extensive *numerical* signal processing, or numger crunching. These systems generally require signal processing al-

gorithms for the detection, identification and tracking of multiple targets against interference environments which may include combinations of unintentional interference, such as RFI or background clutter, as well as intentional interference, such as various ECM jamming threats. Often these systems make use of multi-mode sensors on the same or different platforms to obtain performance improvements over that which can be obtained with a single sensor alone.

The numerical signal processing algorithms are generally developed on the basis of well-defined statistical decision theoretic concepts under specific stochastic modeling assumptions on the nature of the targets, the interference environment and the sensors themselves; including the relative geometries between multiple targets and possible multiple platforms. These numerical signal processing algorithms are typically *optimum*—according to some appropriately defined statistical criterion- – under the specific stochastic modeling assumptions in effect. Unfortunately, these underlying stochastic modeling assumptions are, in many cases, only approximations to reality. Furthermore, there are situations where it's almost impossible to provide *a priori* specification of an underlying stochastic model with any degree of accuracy; e.g., a target with previously unobserved characteristics or a new ECM jamming threat. Finally, there are situations where the appropriate underlying stochastic models change rapidly as a function of time. In situations such as these it's extremely difficult, if indeed possible, to define an optimum processing structure in any meaningful way.

In situations such as described above, where it is difficult or impossible to define *optimum* numerical processing structures, several alternatives have been pursued. These have included the use of *robust* statistical decision procedures, which are less finely tuned to underlying stochastic modeling assumptions, and/or *adaptive* procedures, which attempt, in one way or another, to estimate or learn the underlying stochastic model and adaptively update the processing structure in an effort to "tune" it to the current modeling estimates. Both of these procedures have met with some successes in specific applications. However, it is becoming apparent that a totally stochastic model-based view of the world is somewhat limited. More specifically, in many real- world problems there is often additional information available which is non- numerical and does not lend itself to description in terms of a specific stochastic model. If used, this information can often be quite useful, if not crucial, in obtaining significant performance improve-

ments. This additional information is generally *symbolic* in nature and consists of application-specific world knowledge stored in appropriate knowledge data bases. The description, use and processing of symbolic information of this nature is generally associated with the field of *artificial intelligence* (AI). We will refer to the combination of *numerical* stochastic model-based with *symbolic* knowledge-based processing as *intelligent signal processing.* It is our contention that most real- world engineering problems can best be attacked using a combination of numerical and symbolic processing approaches and are candidates for intelligent signal processing approaches. This is particularly the case for multi-sensor surveillance systems operating in dense target environments and subject to ECM threats.

1.4 Mathematical Model—Single Sensor System

It is useful to first consider a generic single-sensor surveillance system as illustrated in Figure 1.1. In this model the output of the sensor on the k^{th} scan, or frame, is represented by the *MxN intensity array,* F_k. The (i,j) element of $F_k$represents the *intensity* at the corresponding pixel position. It is convenient to think of the array coordinates as specifying a Cartesian coordinate frame relative to wome fixed reference, but this need not be the case. For example, in a synthetic aperture radar (SAR), the pixel positions may represent range- doppler coordinates. Likewise, the *intensity* at a given pixel position need not necessarily represent the actual strength of a return, as in an imaging radar, but might represent, for example, the range to a target in a particular azimuth-elevation (AZ-EL) resolution cell. Thus, the actual form of the intensity array, F_k, may be quite different depending upon the nature of the sensor.

The main functions of the detection/correlation processor are to use the information in an intensity array F_k to provide target declarations, correlate (or associate) target declarations with corresponding trackfile entries, provide target reports to the target tracker and update the trackfile, if necessary, by creating new entries or eliminating existing ones. All access to the trackfile is generally through the trackfile processor where explicit rules are provided for track updates, track initiation and/or track elimination. For the most part these rules have been rather *ad hoc.* This is obviously an area where

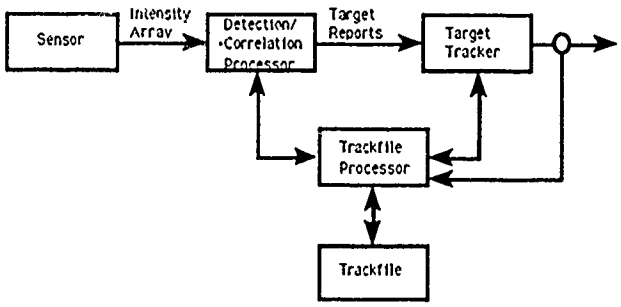


Figure 1.1: Generic Single-Sensor Detection/Tracking Scheme

knowledge-based approaches could be used to advantage, although the exact form this takes remains to be developed.

The function of the detection/correlation processor is to accept the sequence of intensity arrays, $\{F_k\}$, and produce at its output a sequence of target reports, $\mathbf{y}_k^{(i)}$, $i = 1, 2, \ldots, L_k$, where L_k is the number of targets listed in the trackfile at time k. That is, the targets are indexed by track number. This assumes that the track correlation, or association, is performed as an important integral operation within the detection/correlation processor. One method of accomplishing this track correlation is a stochastic model-based approach based upon a generalized Bayesian formulation. That is, the osteriori probability of track association is updated recursively on the basis of accumulated observations which are, in this case, represented by the intensity array F_k, $k = 1, 2, \ldots, L_k$. Like all model-based, strictly numerical approaches, schemes such as this suffer from modelling inaccuracies which are particularly accute in dense target scenarios and in the presence of ECM. This is a situation where intelligent signal processing concepts can be used to advantage. More specifically, if the detection/correlation processor has access to a knowledge database in addition to the pure numerical intensity array and the existing trackfile database, the performance of this track correlation function can be expected to improve considerably. The information may include knowledge about the target characteristics of identified targets, scattering characteristics of local terrain from a terrain database and statistical characterizations of identified ECM threats.

Assuming that track correlation or association has been made, the associ-

ated target reports, or observations, $\mathbf{y}_k^{(i)}$, $i = 1, 2, \ldots, L_k$ are then applied as inputs to the target tracker. A common and versatile approach to modeling target dynamics is to assume that, in some appropriate coordinate frame, the target state, $\mathbf{x}_k^{(i)}$, and associated observation $\mathbf{y}_k^{(i)}$, are described in the familiar form

$$\mathbf{x}_{k+1}^{(i)} = \mathbf{A}_k^{(i)}\mathbf{x}_k^{(i)} + \mathbf{B}_k^{(i)}w_k; \qquad k = 0, 1, \ldots, \tag{1.1}$$

and

$$\mathbf{y}_k^{(i)} = \mathbf{H}_k^{(i)}\mathbf{x}_k^{(i)} + \mathbf{v}_k^{(i)}; \qquad k = 0, 1, \ldots, \tag{1.2}$$

for $i = 1, 2, \ldots, L_k$. Here $\mathbf{A}_k^{(i)}$ and $\mathbf{B}_k^{(i)}$ are possibly time-varying matrices describing the state of the i^{th} target with $\{w_k\}$ an independent and identically distributed (i.i.d.) Gaussian sequence representing uncertainties in target dynamics. Similarly, $\mathbf{H}_k^{(i)}$ is a possibly time-varying measurement matrix with $\{\mathbf{v}_k^{(i)}\}$ an independent, although not necessarily identically distributed, stochastic sequence representing measurement noise.

The target tracker in Fig. 1.1 then processes the noisy measurement data $\mathbf{y}_k^{(i)}$ to produce the sequence of state estimates $\hat{\mathbf{x}}_k^i$ for $i = 1, 2, \ldots, L_k$; i.e., one estimate for each state presently in the track file. A typical approach is to utilize a recursive linear least-mean square (lms), or Kalman-Bucy, filter as the target tracker. In that case, the sequence of state estimates are defined recursively according to

$$\hat{\mathbf{x}}_k^{(i)} = \mathbf{A}_k^{(i)}\hat{\mathbf{x}}_{k-1}^{(i)} + \mathbf{K}_k^{(i)}\tilde{\mathbf{y}}_k^{(i)}; \qquad k = 0, 1, \ldots;\ i = 1, 2, \ldots, L_k, \tag{1.3}$$

where $\mathbf{K}_k^{(i)}$ is the Kalman gain matrix and $\tilde{\mathbf{y}}_k^{(i)}$ is the corresponding *innovation* component. In the absence of any ambiguity in track association this would generally be defined according to

$$\tilde{\mathbf{y}}_k^{(i)} \underset{=}{\Delta} \mathbf{y}_k^{(i)} - \mathbf{H}_k^{(i)}\mathbf{A}_k^{(i)}\hat{\mathbf{x}}_{k-1}^{(i)}; \qquad k = 0, 1, \ldots; i = 1, 2, \ldots, L_k \tag{1.4}$$

In a dense, multi-target environment, however, the innovation component driving the Kalman filter must reflect the possible errors in associating measurements with trackfile entries. For example, suppose that at time k there are J_k measurement reports labelled $\mathbf{y}_k^{(j)}, j = 1, 2, \ldots, J_k$ where J_k can be less than the number of trackfile entries due to *missing* reports or greater than L_k due to *extraneous* reports. These latter *extraneous* reports may be

due to clutter (background noise) or intentional jamming of the sensor. In any case, define

$$\tilde{\mathbf{y}}_k^{(i,j)} \underline{\underline{\Delta}} \mathbf{y}_k^{(j)} - \mathbf{H}_k^{(i)} \mathbf{A}_k^{(i)} \hat{\mathbf{x}}_{k-1}^{(i)}; \qquad i = 1, 2, \ldots, L_k; j = 1, 2, \ldots, J_k. \tag{1.5}$$

Then one approach to defining a *modified* innovation component which reflects our uncertainty in trackfile association is to take

$$\tilde{\mathbf{y}}_k^{(i)} = \sum_{j=1}^{J_k} \rho_{i,j}^{(k)} \tilde{\mathbf{y}}_k^{(i,j)}; \qquad i = 1, 2, \ldots, L_k, \tag{1.6}$$

where $0 \leq \rho_{i,j}^{(k)} \leq 1$ is chosen to reflect our belief that the j^{th} measurement report is associated with the i^{th} trackfile entry. If there are no missing reports or association errors, then for each i, $\rho_{i,j}^{(k)} = 1$ for $j = 1$ (with measurement reports suitably relabeled) and $\rho_{i,j}^{(k)} = 0$, $j \neq i$. In this case Eq 1.6 reduces to Eq 1.4.

An application of AI is to provide methods for formulating and recursively updating the *belief weights*, $\rho_{i,j}^{(k)}$ on the basis of accumulated measurements or evidence. This may include a generalized Bayesian approach as well as knowledge-based AI techniques. The techniques can be evaluated on the basis of the resulting mean-square estimation accuracies.

Regardless of the technique used for track correlation, the belief weights $\rho_{i,j}^{(k)}$ will generally be zero for measurements, $\mathbf{y}_k^{(j)}$, that are sufficiently removed from the *predicted* measurement $\hat{\mathbf{y}}_{k|k-1}^{(i)}$ associated with the i^{th} target on the basis of all previous associated measurements. More specifically,

$$\hat{\mathbf{y}}_{k|k-1}^{(i)} = \mathbf{H}_k^{(i)} \mathbf{A}_k^{(i)} \hat{\mathbf{x}}_{k-1}^{(i)}; \qquad k = 0, 1, \ldots; \; i = 1, 2, \ldots, L_k, \tag{1.7}$$

so that from 1.5

$$\tilde{\mathbf{y}}_k^{(i,j)} = \mathbf{y}_k^{(j)} - \hat{\mathbf{y}}_{k|k-1}^{(i)}; i = 1, 2, \ldots, L_k; \; j = 1, 2, \ldots, J_k. \tag{1.8}$$

Then we would expect that $\rho_{i,j}^{(k)} = 0$ if

$$\left\langle \tilde{\mathbf{y}}_k^{(i,j)}, \mathbf{Q}_k \tilde{\mathbf{y}}_k^{(i,j)} \right\rangle > T_k; \qquad k = 0, 1, \ldots \tag{1.9}$$

where $\langle \cdot, \cdot \rangle$ represents the Euclidean inner product, $\mathbf{Q}_k$ is a positive-definite weighting matrix and T_k is an appropriately defined threshold. For example, if $\mathbf{Q}_k = I$, the identity matrix, then Eq 1.9 becomes: set $\rho_{i,j}^{(k)} = 0$ if

$$\left\| \tilde{\mathbf{y}}_k^{(i,j)} \right\|^2 > T_k; \qquad k = 0, 1, \ldots; \tag{1.10}$$

where $\| \cdot \|$ is the Euclidean norm. In this case we are simply erecting a window about the predicted measurement, $\hat{\mathbf{y}}_{k|k-1}^{(i)}$, and eliminating all measurements from consideration which fall outside the i^{th} window. The criterion of Eq 1.9 behaves similarly, except now the shape of the window is determined by the weighting matrix $\mathbf{Q}_k$. In a sense this is a form of robust filtering where we reject outliers as improbable. It is possible that other robust Kalman filtering techniques may also provid a way to mitigate the effects of large errors due to track association errors. Knowledge-based techniques would also provide ways to identify and exorcise gross observation errors.

For a given window shape and/or size, it is expected that overall performance will depend upon the probability of correctly detecting a target, P_D, as well as the false alarm probability P_F. This last quantity represents the probability of generating a spurious target report due to background noise (clutter) or intentional interference. Based upon previous work on characterizing the various sensors, we believe that new target detection techniques can be developed and incorporated into the detection/correlation processor of Figure 1.1. These detection schemes use the intensity array, F_k, as observables and are characterized by their receiver operating characteristics (ROC's) which represent a plot of the detection probability, P_D, versus false alarm probability, P_F. By developing the appropriate methodology for evaluating tracking accuracy as a function of (P_D, P_F), together with a specified threat scenario, it should be possible to relate overall performance to the detector implementation through the ROC. This approach can be developed on the object-oriented simulator test bed that is to be developed in the follow-on project.

1.5 Mathematical Model—Multiple Sensor Systems

Up to this point we have considered only single-sensor systems. Clearly, the performance of such a system will provide a useful baseline for relative performance evaluation. Our main interest, however, is in multi-sensor systems interconnected via a communication network. It is clear that use of multiple sensors should allow potential improvements in tracking accuracy although, at this point, it is not clear what limiting role the communication network will play. This is most easily investigated by using a test bed such as the one under development. As an example, it can be appreciated that there may be advantages to sharing trackfiles among the various sensor platforms, but it is not clear whether or not there is a benefit in the sharing of data at a lower system level. Would there be a benefit in having sensor platforms share target reports and each create and maintain separate, and presumably similar, trackfiles? Clearly, this may help the target detection/correlation process, but it will require a communication network that can support high data rates.

To provide answers to such questions it is useful to adopt an information theoretic perspective to explore some of the basic tradeoff issues. For example, suppose there are N sensor platforms each sharing target reports over a multiple-access channel. Furthermore, for simplicity, we will assume each sensor platform has access to the same trackfile and there are no track association errors. This is admittedly a gross oversimplification but it does provide a starting point in investigating the effect of the communication network on the overall system tracking accuracy.

The simplified situation under consideration then is illustrated in Figure 1.2. Here, the target reports associated with the i^{th} target at the j^{th} sensor are indicated as $\mathbf{y}_k^{(i)}, i = 1, 2, \ldots, L_k, j = 1, 2, \ldots, N$. Again, the subscript k is intended to represent a time index. Each of these target reports is encoded for transmission over a multiple-access channel. The encoding rate is R_j bits/source-sample for the j^{th} sensor and the correxponding encoder output sequence is indicated as $\left\{z_k^{(i,j)}\right\}$ in Figure 1.2. Letting $\left\{u_k^{(i,j)}\right\}$ represent the channel output sequence associated with the i^{th} target report from the j^{th} sensor, the channel can be described by the transition probability $P(\mathbf{u}|\mathbf{z})$, where $\mathbf{z}$ and $\mathbf{u}$ are N-tuples representing channel inputs and out-

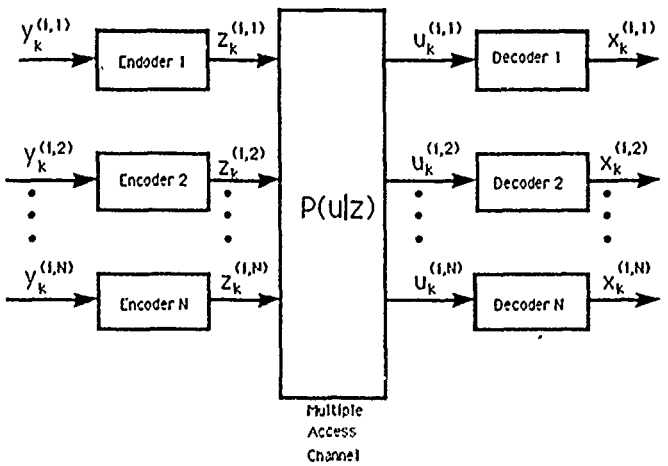


Figure 1.2: Simplified Multiple-Sensor Tracking/Communmication System

puts, respectively. This, of course, assumes a discrete memoryless constant (DMC) channel. The output sequences $\{u_k^{(i,j)}\}$ can be decoded to produce the sequence of state estimates $\{\hat{\mathrm{x}}_k^{(i,j)}\}$.

A more interesting situation, however, is when each of the decoders has access to the channel output vector $\mathrm{u}_k^{(i)} = \left(u_k^{(i,1)}, u_k^{(i,2)}, \ldots, u_k^{(i,N)}\right)^T$ and can produce a *single* state estimate $\left\{\hat{\mathrm{x}}_k^{(i)}\right\}$. Clearly, this should be a better estimate than the local estimate produced at any single sensor or better than the individual estimates $\left\{\hat{\mathrm{x}}_k^{(i,j)}\right\}$ illustrated in Figure 1.2 when only *partial* observation of the channel output is available.

If no encoding errors are made and if sufficient channel capacity is available it is clear that the estimate $\left\{\hat{\mathrm{x}}_k^{(i)}\right\}$ of the preceding paragraph is equivalent to the Kalman-Bucy filter with input measurement vector

$$\mathrm{y}_k^{(i)} = \left(\mathrm{y}_k^{(i,1)}, \mathrm{y}_k^{(i,2)}, \ldots, \mathrm{y}_k^{(i,N)}\right)^T$$

; i.e., the pooled target reports from each of the N sensors. Such a filter may be viewed as a state-augmented Kalman-Bucy filter whose performance may be analyzed in a relatively straightforward manner. However, encoding errors and/or limited channel capacity creates some new issues. For example, suppose we measure performance in terms of the mean-square error

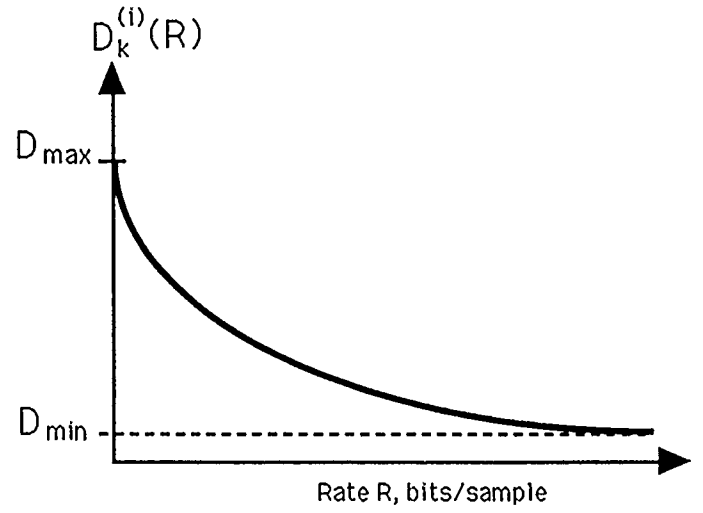


Figure 1.3: Typical Rate-Distortion Function $D_k^{(i)}(R)$

$$D_k^{(i)} = E\{\|\mathbf{x}_k^{(i)} - \hat{\mathbf{x}}_k^{(i)}\|^2\}; \qquad i = 1, 2, \ldots, L_k;\ k = 1, 2, \ldots \tag{1.11}$$

Furthermore, suppose each of the encoders in Fig 1.2is operating at identical encoding rate R bits/source-sample. Then the minimum achievable mean-square error, or distortion, is described by the so-called rate-distortion function, $D_k^{(i)}(R)$. Given a stochastic description of the state evolution and measurement model (cf Eq 1.1 and Eq 1.2, respectively) it is, in principle possible to explicitly evaluate the rate-distortion function $D_k^{(i)}(R)$. A typical plot of $D_k^{(i)}(R)$ is illustrated in Fig 1.3.

Note from Fig 1.3 that as R becomes large, the minimum mean-square error of the Kalman-Bucy filter employing the pooled measurement vector $\mathbf{y}_k^{(i)}$. Achieving this performance, however, requires an infinite rate R or, correspondingly, infinite bandwidth. For finete rate, or bandwidth, the achievable mean-squared estimation error can be considerably larger than D_{min} as illustrated in Fig 1.3. In this case, classical analysis of the mean-square estimation accuracy produces misleading results. An information theoretic formulation, however, clearly identifies the limiting effect of the communication network on overall performance.

We have described an approach to characterizing the limiting performance of an integrated tracking/communication system capable of sharing *correctly* associated target reports. It remains, however, to develop this methodology and to investigate the relative behavior of various suboptimum encoding/transmission schemes. Another topic of interest is to investigate methods for fusing or sharing raw target reports and using the pooled information to aid target correlation or association. Clearly, this requires additional bandwidth and the rate/performance tradeoffs need to be formulated and characterized. Finally, by fusing or sharing the individual sensor intensity arrays, $F_k^{(i)}$, $j = 2, \ldots, N$, it may be possible to aid in the target declaration process and all subsequent downstream processing.

The tradeoff investigations are sufficiently complex for realistic scenarios to make it impractical to do them without the use of a versatile simulation tool. The combination of the simulation tool and the theoretical foundation will provide insight into the practical design decisions which are implicit in the parametric tradeoffs.

Chapter 2

Model of the Radar Environment

To make an intelligent analysis of a radar signal, a controllable yet realistic environment is necessary. This can only be achieved by simulating the complete radar system. This report is a contribution towards modeling and simulation of a surveillance radar system and its environment. A radar system is an electro-magnetic system that extracts information about the objects that reflect or scatter the incident electromagnetic energy towards the receiving antenna. Depending on the minimum receivable power and the radiated power, objects several miles to millions of miles away can be detected. A radar sends electromagnetic signals of certain wavelengths and of specific shapes, and receives the reflected or back-scattered signals that reach the receiving antenna. Using this received signal it is possible to extract the informations like size, location, velocity of the object that scatter it. Given a context(an external knowledge source), the extracted information can be used to classify and recognize the objects.

A pulse-radar transmits a pulse of electromagnetic signal and receives the back-scattered signals for a finite amount of time. This receiving period can be quantized, and the corresponding power distribution in space, at each quantization can be estimated. This quantization can be suitably chosen such that the echoed signals can be reproduced using the samples of back-scattered power from quantized time frames. In this paper we explain an object oriented model to synthesize echoed radar signals through simulation of the radar, the earth surface, the atmosphere and the atmospheric

disturbances, and the targets. We also explain the development of the necessary organization and control structures. To avoid making the scope of our design too restrictive, we have assumed multiple receiver radar system, with a pulse-transmitter(see Fig. 2.1). Once the signal structure is known for the simulated situation, it is possible to infer the target information by studying various correlating and invariant factors of the signal.

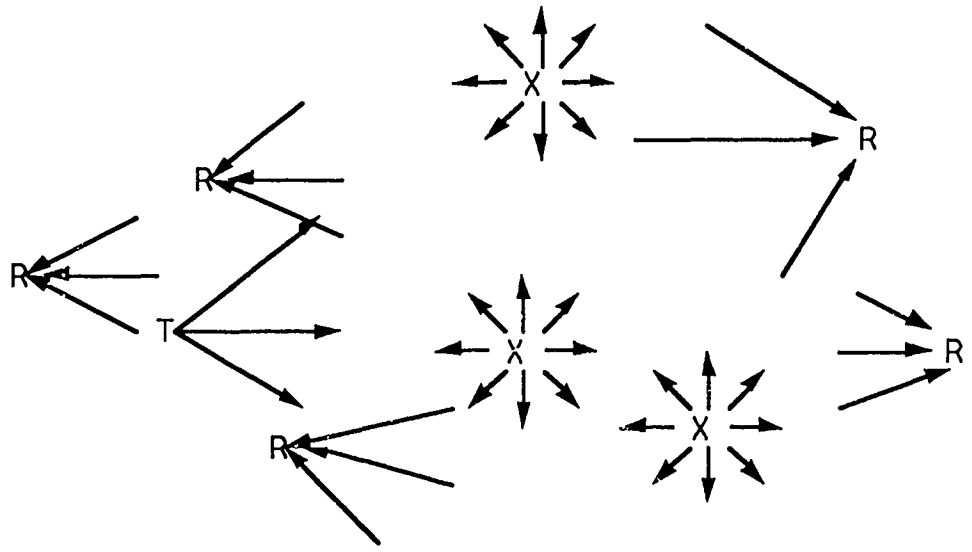
2.1 Overview

A schematic representation of the overall structure of the simulation is shown in Fig. 2.2. This structure has four main parts, viz.

1. Radar Module,
2. Environmental module,
3. Target module.
4. Environment

The links connecting these modules show the flow of information between the modules. Each module receives the necessary specifications from the connected specification module. The radar module is subdivided into transmitter module and the receiver module. The transmitter module provides the transmitted signal objects to the environment at the transmitting antenna. The receiver module samples the back-scattered signal objects from the environment at the receiving antenna. The environmental and target modules provide comprehensive models to simulate the environment. These models are location independent objects instances of which can be activated through proper input specifications. These model instances receive the incident signal objects and output new signal objects which are modified in accordance with their transmission characteristics.

The environment is composed of active instances of signal objects and environmental and target models. The extent of the simulated space is limited by the specified range of the radar. The receiver module samples the power available at the receiving antenna at the end of predetermined time period. The received power represents the back-scattered power. The receiver modulates this power on to an IF(Intermediate Frequency) signal. This IF signal



T - Transmitting antenna
R - Receiving antenna
X - Target

Figure 2.1: Typical relative locations of transmitting & receiving antennas & targets.

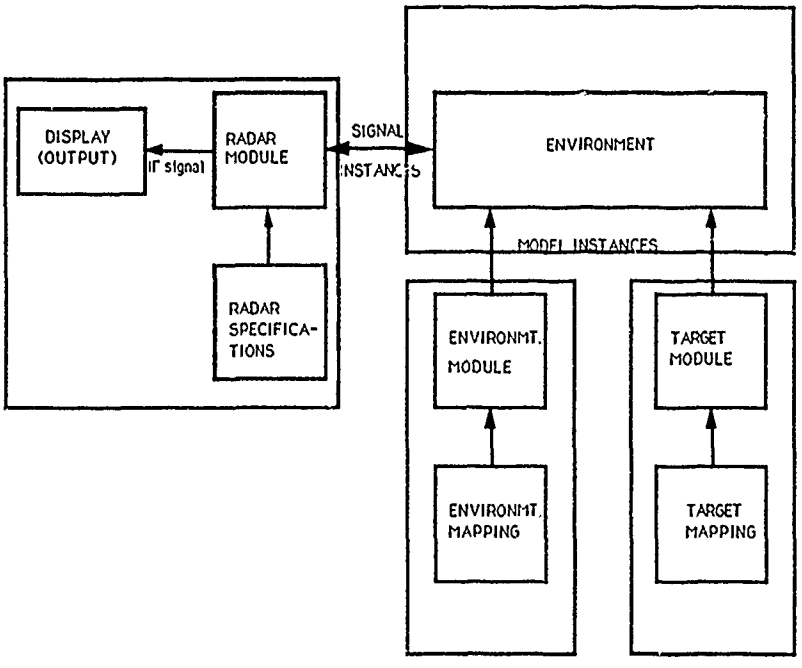


Figure 2.2: The overall structure of the simulation

can be either displayed through the display module or can be used for the analysis of the targets.

2.2 Previous Work

Most of the earlier work is concerned towards development of models for specific environmental and target conditions. As the stress was towards real time simulation, complete radar system simulation was less attractive than using a radar. Now the stress is shifting towards intelligent analysis of the received signal.

2.2.1 Radar System Simulation

An early work towards radar system simulation is [Utsi-77]. In [Utsi-77], a real time radar signal simulator for an aircraft based radar is proposed. The view of the radar is assumed to be 256 X 256 square miles of sea surrounded by land, with many targets in the foreground. The system input is specification of various system parameters. The sea clutter and target scintillations are generated using statistical models. The radar cross sections of targets are precalculated and stored in the EPROM. This storage is used as a lookup table and is available only for discrete orientations. To achieve real time capabilities and to provide accuracy and repeatability, TTL and MOS shift registers are used.

Since transmitted signal is controllable, transmitter can be viewed as a simple channel, which provides the specified signals to the environment, in the direction of the transmitting antenna. But the simulation of a receiver is complex. A good description of the state of the art receivers and transmitters can be found in [Skolnik-84].

2.2.2 Environment Models

There are many publications on the simulation of environmental effects on the back-scattered radar signals. Each of the proposed model, describes a specific section of the environment. In our work, these specialized simulations are used as separate models, and there interactions are governed by the input specification to the environmental model. In general these models can be classified into two subclasses.

1. Statistical models

2. Geometric models

Statistical Models:

One common terrain of importance is the rural terrain. In [Smith 86] various experimental results are presented to justify the proposed *Negative binomial model* for scatter density of rural terrains. The model is developed using a radar terrain imagery. Local scatter magnitude analysis is used to distinguish between the natural and artificial objects. This experimental measurement of density of strong scatterers in a rural terrain are then compared with the negative binomial distribution model. These two data, map better if the local windows selected are smaller in size. A simple model for scattering by rough surfaces is given in [Eftimiu 86]. This model is based on the electric field integral equation. The model assumes that the surface is randomly corrugated in only one dimension. A computationally simple form of correlation function is used.

Geometric Models:

[Chamb 82] provides an evaluation of the two available geometric models, i.e. *Longley-Rice model* and *Geometrical Theory of diffraction* (GTD) model. A new theory of operation for terrain sensitive *propagation path loss model* based on geometrical theory of diffraction, modified for finite conductivity and local surface roughness is also given. The paper describes the theory of operation of each of these models, to expose known deficiencies and to compare their results with the measured data. On the similar grounds SEKE[Ayasli 86], is a computer model for low altitude radar propagation over rough terrain. This is a new site-specific propagation model for general terrain. It makes use of the other similar models like Geometrical optics(GOPT), Low altitude spherical earth (LAPSE), and Low altitude propagation knife edges (LAPKE), to compute multipath spherical earth diffraction and multiple knife edge diffraction losses. The algorithm is selected based on terrain geometry, antenna and target heights and frequency. A comparison of model predictions with measurements over several paths ranging from plane to rough, at five frequencies ranging from X-band to VHF, is presented.

2.2.3 Target Models

In contrast to environmental models, target models are relatively well defined. These models are dependent on the geometric and optical behaviors of the target surfaces in the view of a radar. Normally, targets are charac terized by their impulse responses to the electromagnetic radiation. But it is computationally too expensive to sample the response space, so that necessary amount of information is captured. In [Fok-87], a sampling criteria in the wave number space for generating the spatial impulse response of a finite target is described. A proper choice of canonical confinement for a target can greatly reduce the number of samples required to characterize the target's spatial impulse response. Here, the Shanon's sampling theorem is used to find the minimum amount of data required to reconstruct a target image, assuming that after providing the settling time for impulse response of a target, the signal is time limited.

In [Volak 75], a solution is provided for high frequency back-scattered far field from appendages such as an inlet mounted on arbitrary smooth surfaces. The paper shows the effectiveness of the uniform geometric theory of diffraction (UTD), in computing the scattered fields from complex targets, and also provides iterative techniques to find multiple diffracted ray paths to be used in the application of UTD. These techniques are applicable on both numerically and analytically defined surfaces, such as surfaces of the aircrafts, ships, etc.

In the current research, we develop model to simulate a complete radar system, whose characteristics can be tuned to our requirements by setting the input parameters to suitable values. Many of the existing models can be used in our framework.

2.3 Radar Module

The radar module is made of transmitter, receiver and display modules. The transmitter and receiver modules are involved in the signal simulation process, but the display module is a passive module that is used only for the visual display of the simulated signal.

2.3.1 Transmitter and Receiver

Fig. 2.3 shows the overall structure of the radar module. As can be seen from the figure, the transmitter and receiver modules receive input from two different sources, viz. Radar specifications module, and the Environment. The input specifications module provides system parameters like frequency, peak and average power, polarization of the transmitted signal, coordinates, orientation and band-width of receiving and transmitting antennas, receiving antenna temperature, and the synchronous time to synchronize the receiver and the transmitter. The input from the environmental module is the signals back-scattered from targets and various environmental clutter sources, along with motion induced Doppler shifts.

The receiver and transmitter modules are isolated. Fig. 2.4, shows a schematic diagram of the radar module with separate receiver and transmitter modules. The transmitter channels all its input information to the environment, in the form of signal objects. The internal structure of a signal object is out of the scope of this paper. The receiver module samples the signal objects incident on the receiving antenna at the end of each sampling period. The input is initially filtered. The cutoff frequencies of this filter are determined by the transmitted signal specifications in the specifications module and the receiving antenna characteristics. The filtered signal is then modulated over a IF signal of predetermined frequency. The modulator has two variable parameters, viz. amplitude and frequency. The amplitude parameter is controlled by the input signal where as the frequency parameter is controlled by the relative velocity of the corresponding target. When the velocity component is 0, the frequency is the same as specified IF frequency, but frequency increases if the velocity is positive or if the target approaches the radar, and the frequency decreases if the velocity is negative or if the target is moving away from the radar.

2.3.2 Display

The display module simulates an electronic package that is tuned to the carrier frequency of the input IF signal. The simulation takes care of two specific modes of displays, viz.

1. Single time frame mode,
2. Differential (multiple) time frame mode.

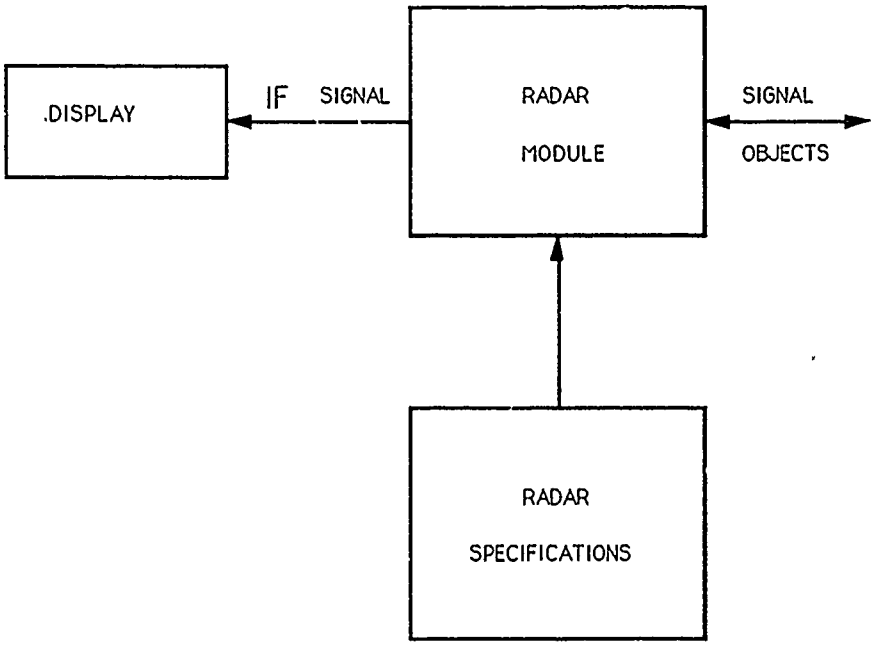


Figure 2.3: The radar module

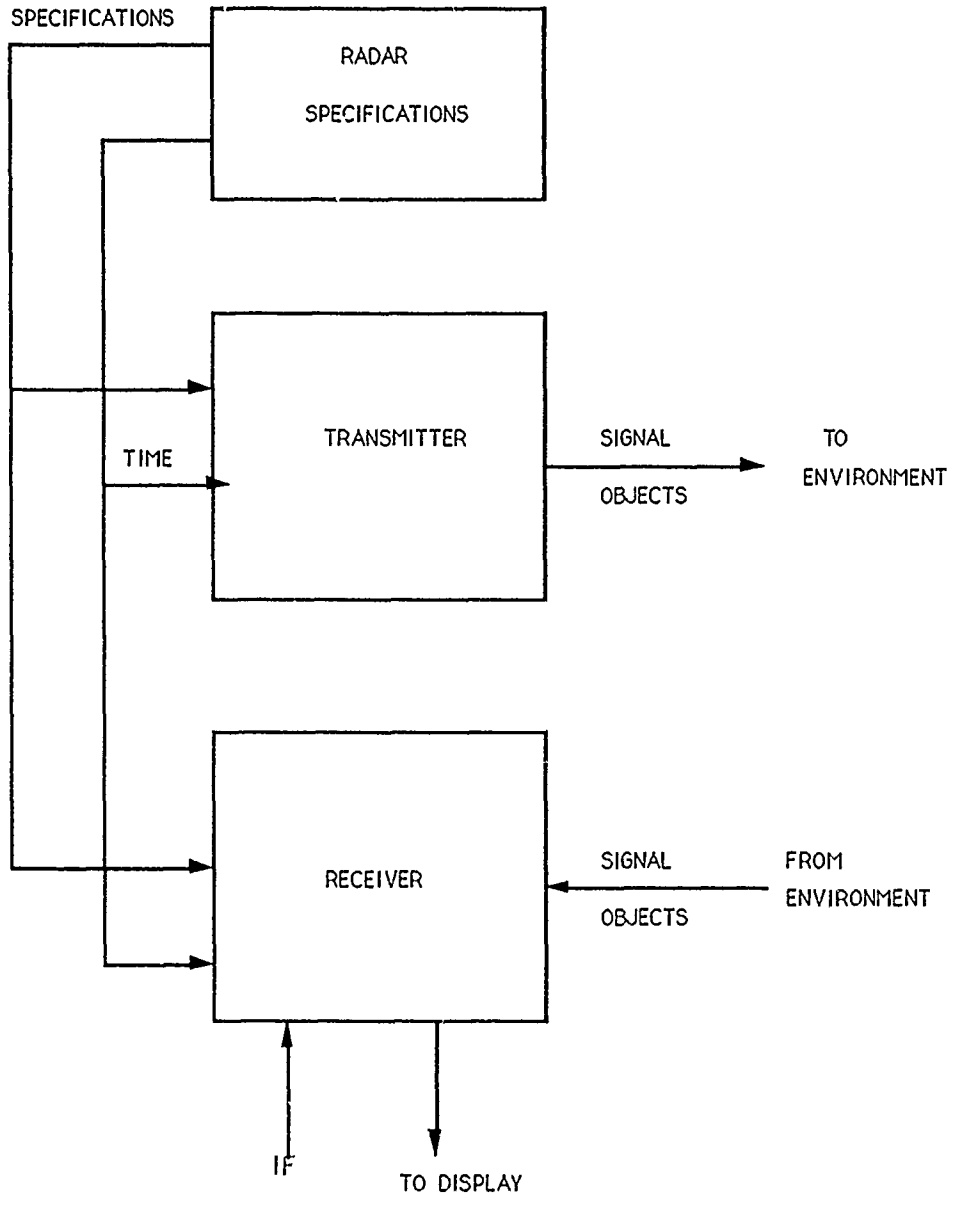


Figure 2.4: Schematic diagram of the radar module with separate receiver and transmitter modules.

The first mode is memory-less, i.e. display at any instant is strictly based on the current input. Here we simulate both AM (Amplitude Modulation) and FM (Frequency Modulation) detectors. The display Doppler shifted signals is done through a FM detector tuned to the IF carrier, and the target location and size information is displayed through the AM detector. This kind of display does not eliminate the clutter information, unless it is run in a differential mode where a difference signal of previous signals with the current signal is displayed. Here the display module stores a specific number of time frames of signals, and the display is made proportional to the phase difference of the successive signals. Since most of the clutter sources are stationary, they do not induce phase difference between signals separated by a finite time, and hence such signals get suppressed in the differential mode. The display is normally used only for manual recognition, and to have a visual feel of signal modifications caused by various specified environmental and target conditions.

2.4 Environmental Module

This module provides existing environmental models as objects, and model parameters are set externally. During the operation, an instance of the specified model is created and made available for the estimation of scattered power at all the specified resolution cells. These instances receive signal objects in the resolution cell to which they belong and output new signal objects which are modified according to the scattering characteristics of the model. A schematic diagram of the environmental module is shown in Fig. 2.5. Since the environmental conditions are stable over short periods, time based transformation of the instances is negligible.

Some of the basic atmospheric factors that modify the transmitted and received signal are described below.

1. Since the beam width of a surveillance radar is fairly wide, in most of the common angles of surveillance, along with the signals reflected from the target, the signals reflected from ground or sea are also received by the antenna. The back scattered signals other than from the targets are the hindrances, and are called clutters. If σ_c is the clutter cross-section, the clutter power C at the receiving antenna is given by

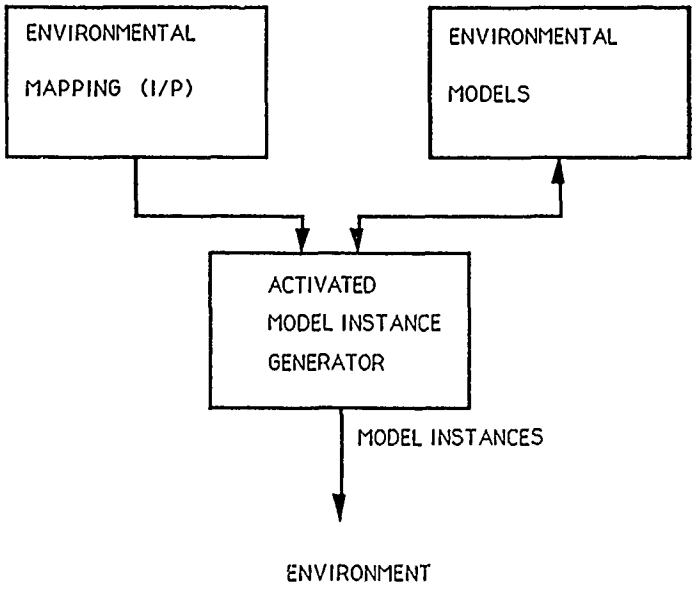


Figure 2.5: Schematic diagram of the environmental module.

$$C = \frac{1}{(4\pi)^2 R^4} P_t G A_e \sigma^0 [0.5 R \theta_B c\tau sec(\phi) - A_p]$$

where

- A_p - projected area of the target on the ground as seen by the antenna.
- P_t is the transmitted power, with antenna gain G and effective aperture A_e
- σ_c, the clutter cross section is substituted by $0.5R(c\tau)\sigma^0\theta_B sec(\phi)$ and σ^0 is surface clutter cross section per unit area.
- R is the range
- c velocity of propagation and τ is the pulse width.
- θ_B and ϕ are the angles as shown in the Fig. 2.6.

If A_p is much smaller than the area of the land that the antenna sees in the background, then it can be neglected.

2. The phenomenon of multiple path propagation as shown in Fig. 2.7, changes the signal power distribution. Hence the back-scattered power received at the antenna can be significantly different. The power received P_r is given by

$$P_r = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} 16 sin^4(\frac{2\pi h_a h_t}{\lambda R})$$

where h_a and h_t are the height of the antenna and the height of the target, respectively.

One can see that power distribution is sinusoidal function of antenna and target elevations.

3. The antenna also receives signals from cosmic sources. Cosmic rays coming from the galaxy and solar rays cause significant amount of noise. These noise sources can be assumed to be black bodies radiating at certain temperatures. T[illegible] temperatures are called noise temperatures or brightness temperatures. Given the equivalent noise temperature of

a cosmic source, the power received by a highly directive antenna is given by

$$P_{cos} = kT_B B$$

- k - Boltzman constant
- T_B - equivalent brightness temperature
- B - Bandwidth of the receiver

For a practical antenna the antenna temperature is defined as the integral of the brightness temperature over all the angles, weighted by the antenna pattern.

4. Signals scattered by precipitation, are also received by the antenna. The scattering by precipitation is a volume phenomenon. Radar cross section per unit volume of precipitation, can be used in calculating the power scattered. This measure is given by the following empirical formula.

$$Z = ar^b$$

- a, b - experimentally found constants, and Z is the radar reflectivity factor.

The constants 'a, b' depend on the type and intensity of precipitation.

5. There is also scattering from birds, insects etc. At times, the scattering because of the air turbulence is also significant. Although scattering from these sources is normally omitted, it should be taken into account if a more accurate and reliable model is desired. Since it is difficult to predict the air turbulence and the movements of birds and insects, the signals scattered from these sources are called angel echos.

6. Besides scattering, the atmosphere also absorbs(attenuates) the signals. This absorption causes a reduction in the intensity of the signal power. The rate of absorption is given in dB/km. The signals are also absorbed by precipitation and moisture.

7. Another important atmospheric factor that affects the radar signals is refraction. The refractive index of atmosphere is normally not uniform; it varies with temperature and pressure in the atmosphere. Refraction causes an object to appear at a place different from its actual location.

Along with the above described global clutter models, specific clutter models are also used for modeling the environment. These models are derived from various published models like GTD[Chamb 82] , SEKE[Ayasli 86], etc. When these models are activated through proper input specifications, their instances are created for the specified parameters and used for the estimation of radar power at any given point in the environment. For example, The model SEKE is based on the assumption that the propagation loss over any path at the microwave frequencies of interest(VHF to X-band) can be approximated by one of the *multipath, multiple knife edge diffraction, or spherical earth diffraction losses or a weighted average of these three basic losses.* The model uses the algorithms developed at Lincoln Laboratory for smooth sphere reflections (GEOSE), multispecular reflections (GOPT), multiple knife-edge diffraction (KEDEY), and spherical earth diffraction (SPH35), as the subroutines. The proper algorithm is selected based on the terrain elevation data for the propagation path, the altitude and the range of the target, and the radar frequency, i.e. the algorithm is selected depending on the input specification.

2.5 Target Module

Target module generates model instances of the specified targets, using a library of scatter models of the basic solid shapes like ellipsoids, planes, spheres, etc. The control structure of a target module is similar to that of the environmental module, but here the most of the models are *time controlled.* i.e. the scattering characteristics of a model changes with time depending on the velocity and orientation of the target. These models receive signal objects and output modified signal objects. Fig. 2.8 gives a schematic diagram of the target module. The target generator submodule generates the instances of target models from the input specifications. To account for the trajectory and orientation changes of a target, orientation and trajectory generators are used. These submodules modify the generated instance of the target model in accordance with the position and orientation at a given time instance

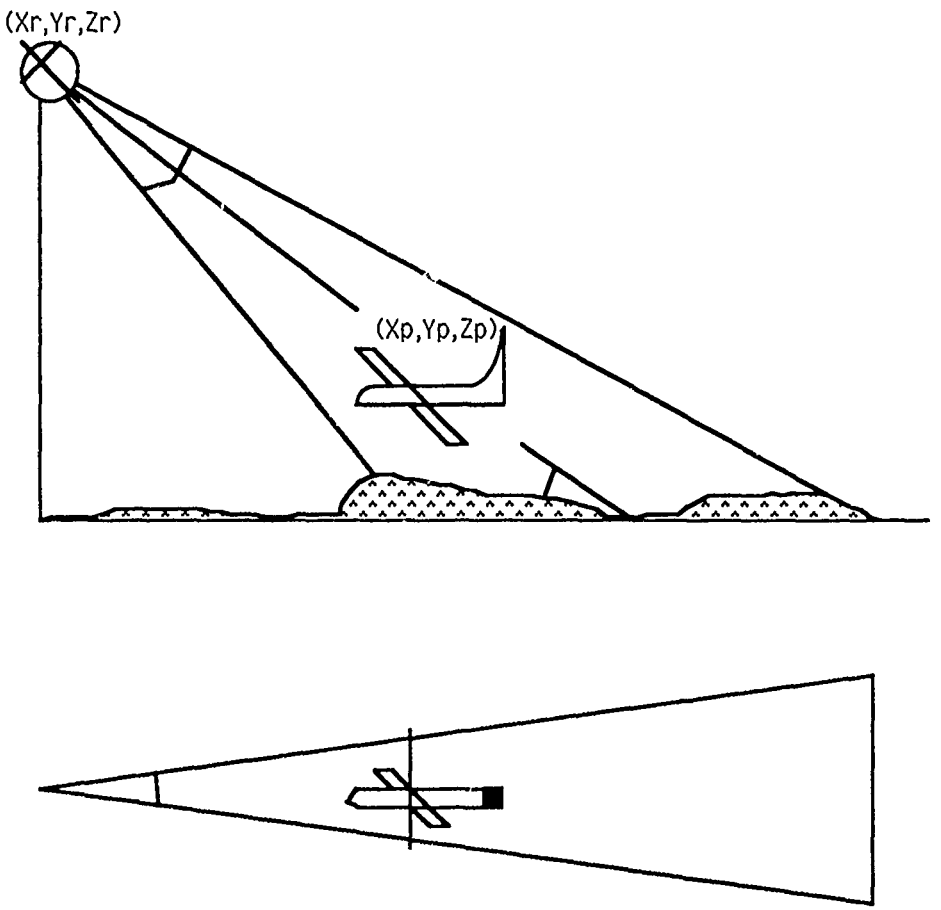
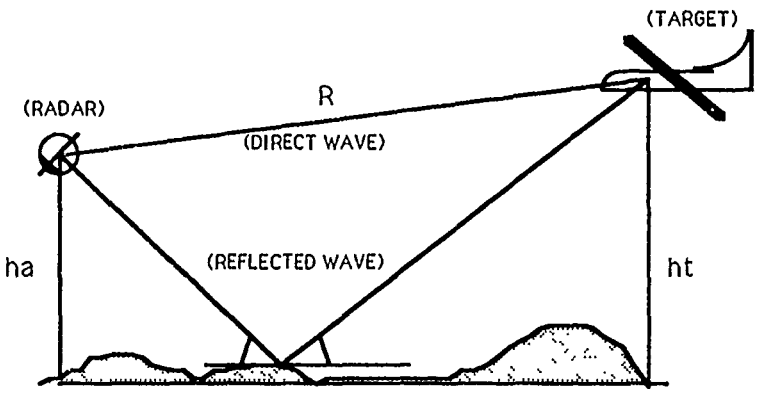


Figure 2.6: Aircraft seen in look-down mode with the earth's surface as background.



1. ha - height of the antenna
2. ht - height of the target
3. R - distance between the antenna and the target
4. - angle at which the indirect path touches the ground

R >> ha

ht >> ha

Figure 2.7: The phenomenon of multiple path propagation

through the transformation module. The trajectory generator returns the position of the target at any given time. The orientation generator modifys the relocated model in accordance with the orientation of the target at that time instance. All the orientations and the spatial displacements are given with respect to a global origin obtained from the simulated environment.

Since many of the targets of interest can be modeled as a combination of finite number of standard geometric solids, the target models can be represented by well-defined mathematical equations. Here, we also make use of many of the existing target models[Fok 87][Volak 85].

2.6 Simulation of the Environment

The task of simulation is to provide a complete radar power distribution map of the space of interest at any time instance. It is not possible to provide continuous time power distribution, and it is not necessary too. It is enough if a snap-shot of the power status is provided at the sampling instance. Secondly, it is neither possible, nor necessary to provide the power status at every point in the space. It is sufficient if the power status is provided at the receiving antenna, targets, and the clutter points. Hence to make the problem computationally feasible, complete signal distribution transmitted or scattered from a scatterer is represented by a signal object. A signal object contains the informations like its origin, time of origin, gain factor of its source(transmitter or scatterer), signal equation and a time dependent decay factor.

The signal objects are input to the environment by the transmitter module. These become available at a scatterer (i.e. environmental disturbance and target) model instances, if there exists a geometrically feasible path from the signal source to the scatterer, and if the required propagation time is elapsed. Initially all signal objects are available to all the scatter models. These model instances modify the signal objects in accordance with their scattering characteristics. Every environmental model-instance that modifys an incident signal, informs the occlusion of the signal to the other environmental and target models that are geometrically occluded to the signal. The occlusion due to targets is quite insignificant. Therefore the target model-instances do not send the occlusion messages. Since at any stage both original and the modified signal objects exist, multiple path and multiple reflection due to the

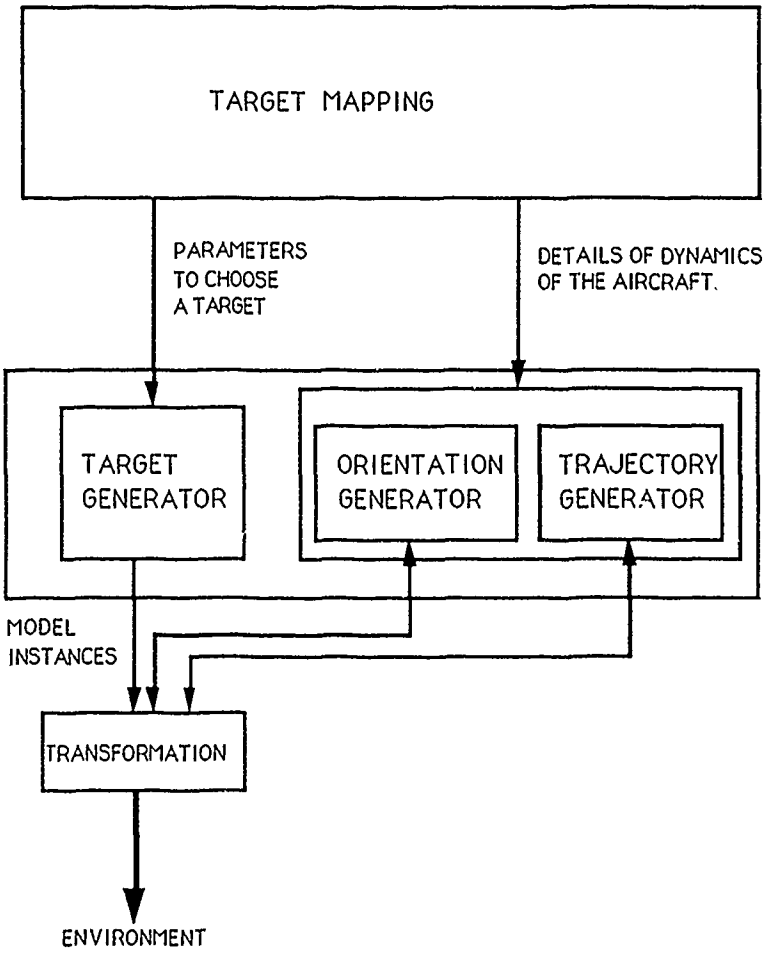


Figure 2.8: A schematic diagram of the target module.

environmental factors are automatically taken into account, but multiple reflection and occlusion due to targets is ignored, but can be easily introduced. Since the signal objects are time dependent, a time dependent decay factor is in-built, and the signal objects disappear after some specified time. At the end of every sampling period, the receiver module periodically samples the available signal objects, at the receiving antenna points.

2.7 Summary

Simulation of a complete radar system is described. This is an object oriented model. The objects are models of specific sections of the environment that are defined independently and are activated by the input specifications. Transmitted signals are represented by objects which are modified by the various atmospheric and target models. The receiver samples the signal objects available at the receiving antenna, at the end of each sampling period. The received power is modulated on a standard IF signal. This IF signal is the simulated radar signal. This is not a real time system, but it is useful in the analysis of various signals in a controlled environment, which is difficult to achieve otherwise.

Chapter 3

Discrete Object-Oriented Simulation

Discrete event simulation (DES) is a computerized technique for experimenting with models of physical systems. It allows one to investigate the impact that changes in individual systems elements have on the performance of the total system. If the simulation system is sufficiently flexible and powerful, it may also promote investigations of different system structures. Both kinds of investigation will be promoted by a simulator tool which makes it easy to model and interconnect system elements.

As an example, consider an investigation of the ability of an automatic pilot to control an airplane in a terrain avoidance application. A basic simulator would require a model of the terrain, the dynamic behavior of the airplane, the senors and the control system. The behavior of the total system would be determined by the interaction of these modules. An improved control system could be tested by substituting its model for the existing control system model in the simulator. So long as the system protocols were met, the simulator should continue to function properly and thereby provide insight about the performance improvement gained by the new controller. One could test the performance of the controller with different aircraft or different terrain by changing those modules and observing the new behavior. Object-oriented tools make it possible to assure that the simulator protocols are met.

in some cases performance can be improved more effectively by changing the system structure than by improving individual elements. As an exam-

ple, it may be possible to improve the detection and tracking capability of a surveillance system either by improving the performance of the existing sensors or by changing the structure of the system so input from other kinds of sensors can be used. Some information that is gathered in another form, perhaps in another time or location, may be available at a much lower price than the cost of improving and retrofitting the existing sensors. The diversity of information sources may provide more useful information and may offer better system performance. However, with a new system would come some new design issues. In particular, it would be necessary to provide ways to integrate the new information into the system control structure. It is therefore necessary to understand the information fusion task and the system restructuring that will achieve that fusion. It would be very useful to be able to add modules to the system and investigate various information fusion and control structures. Object-oriented modeling and simulation offers that capability.

Simulation permits one to investigate the impact of changes in the structure or parameters of a system without the expense or danger of constructing or modifying a physical manifestation of such a system. However, it would be possible to substitute physical elements for some of the modules of the simulation if the proper physical support and interfacing were provided. This would be made easier by using object-oriented modeling and simulation, as discussed below.

Through the use of simulation one may learn about reliability, throughput rates, bottle-necks, response times, and other aspects of system behavior. Simulation can help designers to decide where to reduce or reallocate scarce resources while maintaining or even improving overall system performance.

A general goal for a simulator is to provide an environment and the tools for building and evaluating prototypes of large heterogeneous software and hardware systems. This capability can be used either to evaluate–and understand–existing systems or to design new systems. The designer should be able to quickly build a prototype of a system and evaluate its performance. The tool will then be a tool for creating designs by evolving prototypes. The prototype evolves through successive stages of refinement as the designer gains insight. At any particular stage, the current prototype is an embodiment of the current design. Design by prototyping is a sound, well-understood and cost-effective method. It is being used in modern software engineering tasks to increase both productivity and quality.

3.1 Object-Oriented Programming

Considered by many to be one of the most significant advances in computer science in many years, Object-Oriented Programming (OOP) provides a methodology and the associated programming language support for programming "in the large." The methodology has existed for several years and is the one recommended for Ada system development. Several existing languages support OOP to one degree or another: the oldest language to be considered an OOP language is Simula67, an Algol derivative meant primarily for simulation; newer languages include Ada, C++, Lisp-Flavors and Smalltalk. Other experimental or research languages exist specifically to study OOP include Act1 and Act2.

In an OOP system, the components are considered to be independent objects that interact by sending and receiving messages. An object is an integrated unit of data and procedures, which are called methods, that act on the data. The object is described by state variables, called instance variables, and the data values are used to specify the state by giving values to the state variables. Because objects can interact only by sending messages, the data is encapsulated and protected. A message can contain any kind of information, including data and methods, and may be sent to any number of objects. Messages themselves are objects. Thus, objects may create other objects.

Upon receiving a message an object may process it and take a number of actions. These include modifying its state–which may include "dying," sending one or more messages to other objects and creating new objects. The new objects are created as instances of types of objects that have been described to the system by the programmer. To create an object a message is sent to the system with the specifications for the object and its state.

Objects are grouped into classes which describe the behavior of a kind of object, an instance of the class. This description includes the nature of the internal data and the methods which can be executed. Subclasses may inherit the structure and methods of a class. Object-oriented classes are polymorphic; i.e. the same message can be sent to both a class and its subclasses. For example, if there was a polyhedra class and cube, prism and tetrahedron were subclasses, all the classes could receive a standard message 'print volume' and respond correctly according to the appropriate method for their geometry.

Objects are grouped together in hierarchical classes which can then be put

into separate modules. Parts of the program outside a given class or module can only interact in specified ways with the class or module and thus does not need to know how given methods are executed, variables changed or classes structured. More primitive ideas can be encapsulated in the super classes and thus reduce the level of complexity visible in the subclasses. Information about a particular structure or implementation can be hidden within an object or class.

Programs in object-oriented languages can be readable and comprehensible. The capacity for multi-level reading of simulation programs, focusing on different levels of the class hierarchy or modules, can give new users a quick overview and more experienced users a in-depth look.

The **ESSPRIT** system (Section A) provides program visualization in the form of a system block diagram. The blocks are icons which represent simulation objects on the computer screen. Objects can be added to a prototype by selecting them from a menu, placing them on the screen, specifying their initial parameters and connecting them with other objects. The connections define the paths for messages. Groups of objects can be merged into new icons which represent the complex, so that a hierarchical representation of the system can be built. Various dials, gauges, reports and graphs can be selected to provide reports of activity of any objects in the system.

Such a modular structure can ease modification since one module can be altered without affecting others. This can lead to more flexible and extensible programs. It encourages software reuse since modules from different simulation programs can be pulled in when constructing a new program. A user can easily make changes "on the fly;" new methods can be defined or new classes and objects added. Subroutines and utilities from other languages can also be utilized. This modularity gives the user the ability to pull desirable features into an object-oriented simulation program. Large programs can be broken into many small, independently functioning units.

The hierarchical structure allows the prototype to be developed from top-down. High-level objects, representing major subsystems, can be constructed so that the top-level performance can be evaluated. Once the characteristics of the major subsystems have been stabilized, each of them can be prototyped in terms of its internal building blocks. Particular subsystems can be modeled to lower levels than others–an action which may be useful where some subsystems are taken to be fairly standard while others are more novel. Once the prototype is complete it represents a design for an actual physical

system.

The use of messages and objects seems to be a natural model for many systems. This style of programming parallels the way one intuitively thinks of processes in dynamic systems. Behaviors are attached to specific objects just as the real world entities exhibit different behaviors. Such a software design can also correlate programming objects on a one-to-one basis with real-world objects. It specifies in one place all the data associated with an object and the routines or methods which can manipulate that data. Such structure can allow both naive and experienced users to quickly understand a model.

The object-oriented design is well-suited for concurrent computing on distributed computers. Objects or modules can be placed on different processors and communicate via message passing. The computing environment can contain different kinds of computers and the actual application code on each computer can be in different languages. The only absolute requirement is that the computers respond to messages in the appropriate way, so that there must be a communication and message-handling interface. Existing simulation code or existing data structures can be wrapped in appropriate message-handling shells and used in a simulation environment. This makes it possible, in some cases, to even use existing machine code–a useful option when the source code no longer exists.

The distributed environment can be extended to non-traditional "computers." Physical hardware can be used for particular components in the simulation by providing the hardware and software interfaces to the computer. This makes it possible to use special processors or existing components of systems to speed-up a simulation or to make it work with some real components. The major consideration in using physical devices is their ability to respond appropriately in simulator time rather than in real time.

A special non-traditional "computer" that can be brought into an object-oriented simulation is a human participant. Messages can be sent to the human through the screen and by sound and replies can be sent to the system by the keyboard and mouse. As far as the system is concerned, the human is just another object in the process. This makes it convenient to communicate with the operator, but also allows people to be actual participants in the simulation. The traditional Macintosh display using icons and windows is one special form of object-oriented programming that facilitates communications between the operator and the computer.

The interactive nature of many object-oriented languages facilitates intelligent exploration. The program can be interrupted while it is running, the state of the system can be modified–even by adding or deleting components–and the simulation resumed. The state of the system at the time of interruption can be stored so that it can be restarted from that point to explore a variety of options.

Typical systems that are modeled and simulated will consist of several directly and indirectly interacting components. In the real system–the one being modeled–these components are independent; in the simulation the components should be concurrent or operate in parallel if the computer technology supports it. The control system of the DES model allows each model component to proceed whenever its input is ready. One concern that shows up in the simulation environment but not in the real world is that of time management; a component needs to treat as part of its input the fact that the simulated time is correct in order for that component to proceed.

One of the most important problems with traditional computer programs is their inflexibility. Programs need to be changed–this is their most universal aspect. The older languages and methodologies simply enforce this rigidity, often building highly restrictive descriptions int the functional system requirements. Object-oriented programming directly counteracts that aspect of systems, encouraging early prototyping, with a concomitant lack of details. A broad-brush structure or skeleton is constructed easily and details are added later. When the details change, as they are sure to do, they can be easily modified. In contrast, in traditional programming it is the decisions that are made the earliest that are hardest to change since their affects ripple through the rest of the system.

Since object-oriented programming addresses the problems of structuring a large system– programming in the large–it is a particularly good candidate for scaling up to simulate very large systems.

3.2 Discrete Event Simulation

Discrete Event Simulation is particularly valuable in the design of both hardware and software systems. As noted above, Discrete event simulation (DES) is a computerized technique for experimenting with models of physical systems. It allows one to investigate the impact that changes in individual sys-

tems elements have on the performance of the total system. If the simulation system is sufficiently flexible and powerful, it may also promote investigations of different system structures. Both kinds of investigation will be promoted by a simulator tool which makes it easy to model and interconnect system elements.

The ESSPRIT tool discussed in Section A is intended to provide this kind of environment. It is a simulation shell that permits many different kinds of systems to be simulated. As a part of this project it is being modified to provide a test-bed for multi-sensor radar systems for a multi-target environment.

The multi-sensor multi-target radar environment consists of many directly and indirectly interacting components, as described in Chapter 2. In the real system being modeled these components act independently; in the simulation, then, the components should be concurrent (or operate in parallel if the computer technology supports it). The control system (operating system or run-time system) that directs the overall operation of a DES model permits a component to "run" whenever its input is ready and the simulated time is correct. The concern of time management does not show up in the real world but must be dealt with in the simulated world. A component must treat time as a part of its input so that it may not proceed incorrectly. No component can be allowed to modify the past of any other component.

Object-oriented programming naturally lends itself to Discrete Event Simulation. The objects are the components of the system being simulated. The OOP notion must be modified but slightly: objects send *time stamped* messages to other objects, and the messages must be processed in time-stamped order so that an object can be prevented from modifying the past of another object.

3.2.1 Time Management

In DES there is generally a global clock that gives the simulated time of the system. The primary control is to iteratively wait until the system is quiet ("quiescent"), determine the least time value for the components that are prepared to work, advance the global clock to that time, and start the corresponding process. This is a contrast with continuous simulation in which the clock is stepped by uniform amounts Δt and the simulator integrates systems of differential equations equations.

In simulations such as a radar environment, different components may operate on widely-varying time scales, from microseconds to hours or days. This wide range makes it difficult to run such simulations with continuous simulators since all equations must be stepped by the smallest time step needed by any component. In DES it is possible to allow only those elements which require processing at each time to proceed, so that there can be a gain in efficiency by orders of magnitude.

Continuous simulation systems put the dependencies between components in the model and DES puts them in the component descriptions by describing component behaviors. In that way DES is much more flexible and modular.

A typical operation in DES may go as follows:

1. A client seizes a resource

2. The client uses the resource for N seconds

3. The client releases the resource

The second step causes the particular process to be "put to sleep" for N simulated time tics; i.e., a wakeup is scheduled for $time = currenttime + N$. (The choice of N often depends upon a pseudo-random number generator to express the analyst's choice of a statistical distribution.)

The wakeup manager plays a role similar to an operating system scheduler or a dataflow system monitor (or imagine a hotel's wake-up service). The principal data structure, which operates like a priority queue, is called the "(future) events queue." This events queue is a set of pairs, $(time, event)$; elements can be added to this set without any particular discipline (except that the *time* component can not be less than the current simulated time), but every time a pair is removed, it is the pair with the lowest *time*. One may picture this as either a linked list or as a sorted array, but alternatives that are more effective will be mentioned below in Section 3.2.3.

When a system component "goes to sleep," one imagines the ordered pair, above, "waiting" as a surrogate for the component in the wakeup manager's queue. (In a similar arrangement a surrogate for a process will wait in a queue—waiting, say, to use a particular resource.)

3.2.2 Miscellaneous Issues for DES

3.2.2.1 Queues

Whenever a client attempts to seize a resource that resource may be in use. The client may be able to test for that possibility, but often there are no alternative actions possible (or planned), so the client must "wait" in a queue. Such queues can be either explicit, with names and having statistics gathered and reported, or implicit components of the resources, simply managed by the system but otherwise invisible. In the latter case, the clients may keep track by themselves of the amount of valuable work accomplished over a period of time an the time lost waiting in queues. Generally queues are FIFO, but other structures are possible as well:

1. Priorities of the clients may affect waiting orders.
2. Queues may have a limited capacity and be unable to accept entries beyond a specified limit.
3. Clients may be able to exit from a queue after waiting but being unable to get the service (imagine a client having a finite amount of patience).
4. A queue itself may throw out clients that have waited too long.
5. A queue may time out.

3.2.2.2 Reports Generated

A simulation is run in order to gain some insight into the behavior patterns of a system, so there must be some well-planned output. **ESSPRIT** provides for a range of screen dials, charts, graphs and reports as well as the capability to generate archival reports of the activities of all objects in the system. This capability is being included in the radar simulator.

3.2.2.3 Random Numbers

Pseudo-random number generators that are tunable and trustworthy need to be used. Unfortunately, these are not easily available off-the-shelf. Not only do horror stories abound (one random number generator was found that only output zero, for instance) but there are constantly articles in the

professional literature about how bad they are and how to do it right. The **ESSPRIT** simulator provides the capability to easily incorporate a suite of random number generators and to add a new generator whenever another may be required.

3.2.3 Scaling Up: Time Management

When small systems are scaled up to larger ones, there are bound to be surprises; nonlinearities abound. The problem that often hits simulation systems badly is the management of future event queues. These queues must support an *ordered set* so that whenever an item is removed from the set it is the one with the smallest time value. The obvious way to do this is to keep the list sorted and use an internal form such as a linked list or an array. The cost to enter an item into such a list whose size is M items is proportional to M—it is of time complexity $O(M)$. The cost to remove an item is independent of M—it is of order $O(1)$. Alternatively, one could keep the internal data structure unorganized, but then the cost to enter would be $O(1)$ and the cost to remove would be $O(M)$. The data structure that we suggest here is that used in the algorithm "heap sort." It is an array that is sorted only according to a very sparse constraint rule; namely, instead of requiring that $A_i > A_{i-1}$ for all i such that the expressions are valid subscripts, we require only that $A_i > A_{i/2}$. The heap sort idea, with its fewer constraints, is much easier to maintain. If M denotes the set's current population, then the cost to add a new element to the set is only $O(\log_2 M)$, and the cost to remove an item is also $O(\log_2 M)$. This is particularly striking in its effect: $\log_2 1,000$) is approximately 10 and $\log_2 1,000,000$ is approximately 20.

3.2.4 Scaling Up Using Multiple Processors

As computers are reaching their ultimate limits as imposed by the speed of light and quantum mechanics, the obvious approach is to employ systems designed to do many things at the same time. Simplistically, N computers should be able to solve a problem in $1/N$ of the time a single computer would take. This is simplistic because N systems must generally cooperate, and the cost of coordination and communication can be prohibitive. As we shall see below, the problems involved in distributing discrete event simulation seem to as challenging as any in current computer science research. Several good

ideas are currently under investigation, but the general problem is far from solved.

There are two simple, quick and dirty suggestions to scale simulation systems up using multiple processors. The first is to use the observation that perhaps 80% of the time spent in typical large simulations involves what we have called the "miscellaneous" issues of simulation: time management, statistics gathering and reporting, and pseudo random number generation. These system components have very clear interactions with all the other components of a simulation, and separating them to other processors can easily be accomplished. The issues involved are the same as any other easy decomposition of a problem into parallel components.

The second easy approach is to design a systems simulation as a continuous event system. In this way, a global clock can broadcast the current simulated time to all components, the components can perform an increment of work, and parallelism can be achieved.

To distribute the objects of a simulated system over several processors, the current research suggests that the system needs to lose its global clock, and that every component (or processor) maintain its own local clock. The clocks can then be advanced, each independently of the others. The problem that must be avoided is for one component to change another's past. This is interpreted, in the local-clocks approach, as prohibiting a message of the form (m, t), where t is the time stamp, directed to component C in case component C has already issued a message of the form $(m', t'$ where $t' > t$.

Several methods have been suggested by researchers to avoid the problems associated with local clock coordination: roll-back, massive time messages, and (possibly) restricted time messages.

In case the system reaches a state where the violation of the time ordering of messages must occur, as when a component does receive a message "in its past," a message whose time stamp has a lower time than one that had been sent out by the receiver, then the receiver must initiate work to modify the advances made by the system in the time after the time stamp on the offending message. The simplest way to achieve this roll-back is via a checkpoint-restart procedure (familiar from the attempts to keep a computation going well beyond the mean-time-to-failure for a system.) This entails writing the entire state of the system to sa disk file periodically with the expectation that the system may have to revert to that state and make alternative decisions. An alternative to this sort of overhead is a more selective roll-back

wherein the component, whose past was detected to have been changed by the message with the low time stamp, attempts to revoke messages that are still in the unsent queue (like catching the mailman before he delivers something you wish you had not sent) and sending retraction messages to those recipients that had received the messages that it had tried to cancel. Such a effort could obviously get out of hand, with an ever increasing circle of messages and processes trying to back out of a contradictory situation.

Alternatively, a system monitor could try to detect the occurrence of such problems and keep them from occurring. The research in this direction seems to indicate an equally unacceptable amount of system overhead. One still does not gain substantial advantage from the parallelism.

The final method suggested is for system components to broadcast time coordination messages to the other components regarding its intention to update its local clock. This method also indicates a huge amount of message traffic, but it has suggested in its wake the following method: whenever a component wants to update its clock, it sends messages to the components that can send messages directly to it, essentially requesting permission to update the time. The requesting component elicits promises from those who send it messages that they will not send a message time stamped before a certain time. In order to answer such a request, these other components need to propagate the request backwards through the system to the components that might send messages to them. Although this leads to interesting problems dealing with loops and parallel paths in the message graph, there seems to be a glimmer of hope that the overhead might not cancel out the multiple-processor gains.

Chapter 4

Status

The contract required the accomplishment of four tasks:

1. Describe the multi-sensor surveillance environment in a manner which facilitates the transfer of techniques developed in other applications to its domain.

2. Identify the intelligent signal processing techniques which may be applied to the tasks of identification, sensor fusion for multiple targets and multiple sensors and intelligent tracking.

3. Identify concepts and techniques from other areas of artificial intelligence that will be required to provide the desired system performance.

4. Provide a road map for the development of system elements and a plan for integrating them into a functional body.

These tasks have been accomplished and are summarized in the following sections.

4.1 Task 1

The multi-sensor radar environment has been described in Chapter 2. All of the elements, including radar subsystems, targets, interfering objects and signals are to be modelled as objects in a radar-target domain. This description of the environment facilitates a modular structure in which each element

can be separately modeled. A hierarchy of detail is possible in which certain objects are modelled with more precision than others. As environment modules are developed or refined they can be included in the object-oriented environment.

4.2 Task 2

Intelligent signal processing techniques depend upon the use of a symbolic representation of the information in the signals. The relationships between symbolic processing and traditional statistics-based processing are described in Section 1.3. The symbolic representations can be used to build a knowledge structure which supports sensor fusion. This facilitates the combination of information from many sources for the purpose of target identification and tracking.

4.3 Task 3

Other areas of artificial intelligence that would apply to this problem include plan recognition, evidence combining, threat assessment as well as methods for system control, knowledge-maintenance and truth maintenance. All of these techniques depend on the existence of a symbolic structure for the information that is gathered from the environment. The object-oriented representation of the system, targets, signals and environment facilitates the use of blackboard-like control structures, which have proven to be the most fruitful structures in other areas of intelligent signal processing, including speech recognition and computer vision. These areas cannot be explored without a suitable simulation environment, which is the goal of the next project phase.

4.4 Task 4

The project has identified the object-oriented approach as being the appropriate method to represent the radar domain. The next phase of the project is the development of the simulator structure. This is being done by specializing the **ESSPRIT** simulator shell for the multi-target multi-sensor radar

domain. This then provides the test-bed on which prototypes of sensor fusion systems can be built. As discussed in Section 3, the resulting prototypes are equivalent to system specifications.

4.5 Additional Research

Additional research is required in several areas. This can be carried out in a modular fashion and used to produce object descriptions or prototypes within the simulator environment. In this way systems can be compared and improved by competitive refinement and theory can be evaluated in an experimental domain. Some particular investigations include:

1. Develop a general model for signal objects.
2. Develop models for selected types of sensors.
3. Develop models for clutter and noise.
4. Develop models for jammers and ECM devices.
5. Develop models for knowledge structures which support sensor fusion by implementing the detection/correlation processor function.
6. Develop methods to maintain target track knowledge in a multi-sensor environment with multiple targets.
7. Investigate the effects of finite bandwidth between sensors in multi-sensor fusion systems.
8. Investigate knowledge structures for sensor fusion in a noisy, uncertain environment for target identification or target tracking.
9. Investigate methods to scale up parallel processing systems to achieve maximum performance.
10. Investigate methods to combine non-sensor knowledge or information, such as knowledge of tactics or intelligence reports of enemy activity.
11. Investigate methods to provide real-time situation assessment to field officers in a tactical environment.

The above list is only partial, but it illustrates the range of investigations that can be supported by *combining* AI techniques with modern simulation and prototyping tools. The combination will permit the results of diverse investigations to be built into a framework which can be used to carry out experiments and further investigations. The approach provides an open-ended tool for system modeling, analysis, and design.

Bibliography

[**Ayasli 86**] S.Ayasli, "SEKE: A Computer model for low altitude radar propagation over rough terrain," *IEEE Trans. on Antennas and Propagation,* Vol.AP-34, No.8, pp.1013-1023, August (1986).

[**Chamb 82**] K.A.Chamberlin, R.J.Luebbers, "An Evaluation of Longley Rice and GTD propagation model," *IEEE Trans. on Antenna and Propagation,* Vol.AP-30, No.6, pp.1093-1098, November (1982).

[**Eftimiu 86**] C.Eftimiu, "Scattering by rough surfaces: A simple model," *IEEE Trans. on Antennas and Propagation,* Vol.AP-34, No.5, pp.626-630, May (1986).

[**Fok 87**] F.Y.S.Fok, J.D.Young, "Space frequency sampling criteria for electromagnetic scattering of a finite object," *IEEE Trans Antenna and Propagation,* Vol.AP-35, No.8, pp.920-925, August (1987)

[**Hendry-89**] endry, Barbara. "Distributed Object-oriented Discrete Event Simulation," Master of Science Thesis Proposal, School of Computer Science and Technology, Rochester Institute of Technology, Rochester, NY. April 28, 1989.

[**Misra-86**] isra, J. "Distributed Discrete-Event Simulation." *Computing Surveys* **18**, No. 1, March, 1986.

[**sher-89**] her, David, Shriram Revankar and Ramadoss Venkatesan, "An Object-Oriented Model for a Radar System Simulation," 20^{th} *Pittsburg Conference on Modeling & Simulation*, Pittsburgh, May 5 & 6, 1989.

[**Skolnik-80**] M.I.Skolnik, *Introduction to Radar Systems,* McGraw Hill Book Company, Second Ed. (1980)

[Smith 86] F.W.Smith, J.A.Malin, "Models for radar scatterer density in terrain images," *IEEE Trans. on Aerospace and Electronic Systems,* Vol.AES-22, No.5, pp.642-647, Sept.(1986).

[Utsi-77] V.Utsi, "Real Time Radar Signal Simulator," *International Conference on Radar-77,* pp 478-481 (1977).

[Volak 75]

[Vannicola]Vannicola, Vincent C. and Jack A. Mineo, "Applications of Knowledge Based Systems to Surveillance,"Surveillance Directorate, Rome Air Development Center, Griffis Air Force Base, NY 13441.

J.L.Volakis, W.D.Burnside, L.Peters,Jr., "Electromagnetic scattering from appendages on a smooth surface," *IEEE Trans. on Antennas and propagation,* Vol.AP-33, No.7, pp.736-143, July (1975).

Appendix A

The ESSPRIT Simulation Environment

The **ESSPRIT** (Explorer Simulation and Signal Processing system from the Rochester Institute of Technology) system developed at RIT Research Corporation is a software package written for the Texas Instruments Explorer Lisp Machine. **ESSPRIT** combines an object-oriented approach to simulation with a visual programming interface to provide high-level design capabilities along with the capacity for intelligent exploration of complex systems. Models which contain a large number and variety of interacting entities can be rapidly prototyped using this tool. **ESSPRIT** is presently being used at RIT Research Corporation to develop simulations in manufacturing, system dynamics, and radar design.

A.1 Visual Simulation

Traditional simulation languages require programmers to write large amounts of procedural code to model systems. Typically, an analyst using one of these languages develops a conceptual model of a system, which is passed on to a programmer who writes a program to implement the model. The analyst then receives the results of the simulation execution in the form of reams of statistics. The **ESSPRIT** system allows an analyst who is not an expert in simulation to interactively design, execute and analyze models of complex systems using a graphical interface. The topology of the objects and

transactions involved in a simulation is described using graphical tools. During execution, transactions are animated and selected statistics are displayed in graphs and gauges. At any time the simulation can be interrupted and altered. The result is a highly interactive system which allows an analyst to consider alternative hypotheses, modify models, and visually observe the dynamic behavior of a simulation.

A.2 Object-Oriented Framework

ESSPRIT is implemented using an object-oriented programming system. All data, programs, commands, and even the user are viewed as objects. Each object is an instance of a class, and each class has an associated collection of operations which can be applied to the object. The classes are arranged in a hierarchy so that subclasses can inherit properties of their parents. This approach provides for a high degree of code reusability and encourages the use of data and control abstraction.

A.3 System Architecture

ESSPRIT is composed of three major components: a graphical configuration editor, for designing simulations; a simulation executive, for running simulations; and libraries of simulation objects.

A.3.1 Configuration Editor

An **ESSPRIT** simulation is specified by drawing a block diagram of the system. Icons which represent instances of simulation objects are placed on the screen. An object can be constrained to communicate with another object by drawing an arrow to indicate a message path, or it may communicate with other objects in the simulation in a more generic manner. Each object contains parameters which may be adjusted before or during execution of the simulation. A variety of displays and gauges which monitor the state of the system or individual objects can be selected interactively. Any **ESSPRIT** diagram may be composed into a single object, which can then be used within another diagram. This allows a complex simulation to be con-

structed in a modular, hierarchical fashion, and provides the user with a way of investigating the behavior of a system at different levels of complexity.

A.3.2 Simulation Executive

ESSPRIT uses a process interaction approach to discrete-event simulation. Each object has a process which describes the sequence of operations through which the object passes during its life within the system. An object can be delayed either unconditionally for a certain period of time, or conditionally until a certain condition exists. The actual computation in an object's process may range in complexity from a simple Lisp expression to a large database query or expert system. The simulation executive allows objects to be run either sequentially or concurrently, and controls any animation and displays which are selected.

A.3.3 Simulation Object Libraries

The **ESSPRIT** configuration editor and simulation executive together form a "shell" which contains generic knowledge about simulation, but no specific knowledge about any applications. This domain-specific knowledge is contained in separate application libraries of objects and rules. In this way, a non-expert can build models composed of pre-defined parts while more sophisticated users can create their own objects or modify existing objects.

A.4 Hardware

Currently, **ESSPRIT** simulations run on a TI Explorer Lisp machine, which is a dedicated single-user system intended primarily for use in developing artificial intelligence applications. These computers are typically configured with 8-32MB of memory and a pair of 140 MB disk drives, and cost around $50K. The **ESSPRIT** executive is in the process of being modified to run simulations on a network of Explorers. Each object in a simulation will then be able to exist on any machine on the network. In addition, display and control interfaces to simulation objects may be spread over multiple machines. The **ESSPRIT** system is also being ported to a Macintosh II using Allegro Common Lisp.